Using Web Catalogs to Locate and Categorize Unknown Furniture Pieces in 3D Laser Scans

Oscar Martinez Mozos*, *Member, IEEE,* Zoltan-Csaba Marton*, *Member, IEEE,* and Michael Beetz, *Member, IEEE,*

***Abstract*—In this article, we address the problem of exploiting the structure in today's workplace interiors in order for service robots to add semantics to their sensor readings and to build models of their environment by learning generic descriptors from online object databases. These world models include information about the location, the shape and the pose of furniture pieces (chairs, armchairs, tables and sideboards), which allow robots to perform their tasks more flexibly and efficiently.**

To recognize the different objects in real environments, where high clutter and occlusions are common, our method automatically learns a vocabulary of object parts from CAD models downloaded from the Web. After a segmentation and a probabilistic Hough voting step, likely object locations and a list of its assumed parts can be obtained without full visibility and without any prior about their locations. These detections are then verified by finding the best fitting object model, filtering out false positives and enabling interaction with the objects.

In the experimental section, we evaluate our method on real 3D scans of indoor scenes and present our insights on what would be required from a WWW for robots in order to support the generalization of this approach.

I. Introduction

We expect the future World Wide Web to include a shared web for robots, in which they can retrieve data and information needed for accomplishing their tasks. Among many other information, this web will contain models of robots' environments and the objects therein. Today's web already contains such 3D object models on websites such as Google 3D Warehouse or catalogs of online furniture stores.

In this article, we investigate how autonomous robots can exploit the high quality information already available from the WWW concerning 3D models of office furniture. Apart from the hobbyist effort in Google 3D Warehouse, many companies providing office furnishing have already modeled considerable portions of the objects found in our workplaces and homes. In particular, we present an approach that allows a robot to learn generic models of typical office furniture using examples found in the Web. These generic models are then used by the robot to locate and categorize unknown furniture in real indoor environments as shown in Fig. 1.

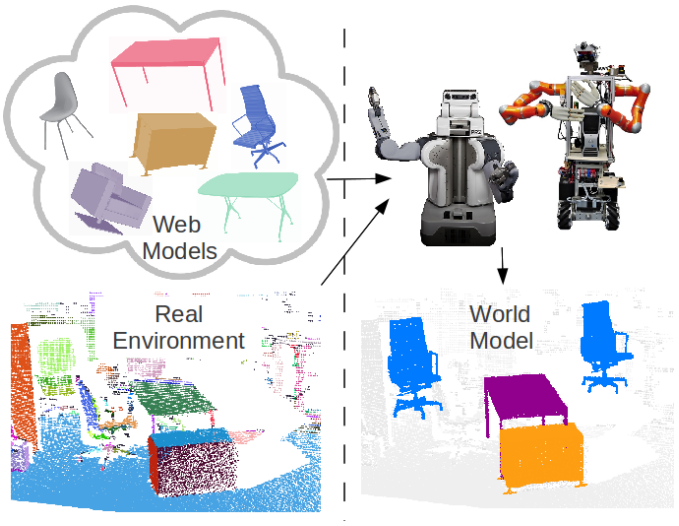


Fig. 1. Using furniture models from the WWW together with a segmented scan of its real environment, the robots create a world model. See the color legend in Fig. 4 (different parts are indicated by random colors).

Furniture pieces share many common parts, especially if their purpose is similar. For example, most chairs have an approximately horizontal and vertical part, and rectangular planar patches are quite common. Thus the key idea of this article is to represent the object models learned by the robot using a vocabulary of these common parts, together with their specific spatial distributions in the training objects.

During the training process the CAD (Computer Aided Design) models from furniture Web catalogs are converted into point clouds using a realistic simulation of laser scans. These point clouds are segmented and the resulting parts from the different training objects are clustered to create a vocabulary of parts. In the detection step, we match similar parts and apply probabilistic Hough voting to get initial estimates about the location and categories of objects found in the scene. Finally, these detections are verified by finding the CAD model that best fits to the measurements. This last step allows the robot to reject false positive detections.

Using larger regions (instead of points) as basic units for learning offers many advantages. As was already shown in [1], the knowledge about different functional units of objects can contribute significantly to a correct detection. Moreover, we use training examples that are similar but not necessarily the same as the objects encountered during the operation of the

* denotes equal contribution.

Oscar Martinez Mozos is with the Laboratory for Intelligent Robots and Vision Systems, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan omozos@irvs.is.kyushu-u.ac.jp

Zoltan Csaba Marton and Michael Beetz are with the Intelligent Autonomous Systems, Technische Universität München, 85748 Munich, Germany {marton,beetz}@cs.tum.edu

robot, thus improving the generalization of the final classifier.

II. Related Work

Although appearance-based object identification works reasonably well using a variety of techniques, the robustness and scalability of many perception systems remains an open issue, as identified by Kragic and Vincze [2]. Ideally, a robot should be able to recognize thousands of objects in a large variety of situations and additionally detect their poses. We will review some of the steps taken in this direction and contrast them to the method proposed in this article.

A widely used technique to recognize objects in point clouds involves local descriptors around individual points. For example, the spin image descriptor [3] is used by Triebel *et al.* [4] to recognize objects in laser scans, and it is also used by de Alarcon *et al.* [5] to retrieve 3D objects in databases. More recently, Steder *et al.* [6] presented the NARF descriptor which is well-suited for detecting objects in range images. Other works apply relational learning to infer the possible classification of each individual point by collecting information from neighboring points. In this sense, Angelov *et al.* [7] introduce associative Markov networks to segment and classify 3D points in laser scan data. This method is also applied by Triebel *et al.* [8] to recognize objects in indoor environments. All previous methods utilize individual 3D points as primitives for the classification, whereas we use complete 3D segments or "parts" represented by feature vectors. We believe that parts are more expressive when explaining objects.

For the detection of complete objects, for example 3D shape contexts and harmonic shape contexts, descriptors are presented in the work by Frome *et al.* [9] to recognize cars in 3D range scans. In the work by Wu *et al.* [10], shape maps are used for 3D face recognition. Haar features are used in depth and reflectance images to train a classifier in the work by Nüchter *et al.* [11]. In these works, objects are detected as a whole, whereas we are able to detect objects by locating only some of their parts, which results in better detections under occlusions and when using different viewpoints. Our work shares several ideas with the approach by Klasing [12], which also detects objects using a vocabulary of segmented parts. However, we apply the classifier directly to the point cloud without looking for isolated objects first.

Part-based object classification in 3D point clouds has also been addressed by Huber *et al.* [13], using point clouds partitioned by hand. In contrast, we partition the objects in an unsupervised manner. Ruiz-Correa *et al.* [14] introduce an abstract representation of shape classes that encode the spatial relationships between object parts. The method applies point signatures, whereas we use descriptors for complete segments.

Many of the techniques in our approach come from the vision community. The creation of a vocabulary is based on the work by Agarwal and Roth [15], and its extension with a probabilistic Hough voting approach is taken from Leibe *et al.* [16]. Voting is also used by Sun *et al.* [17] to detect objects by relating image patches to depth information. Basing our approach on geometrical information allows us to have a single 3D CAD model of an example object in the WWW database, since the different views can be generated by the robot. Combinations of 3D and 2D features for part-based detection would definitely improve the results [18].

For matching problems, RANSAC and its variations are widely used due to the flexibility and robustness of the algorithm [19], [20]. To register different views of an object, local tensors are applied by Mian *et al.* [21]. Moreover, Rusu *et al.* [22] limit the point correspondences by using local features.

Finally, using synthetic data for training data is an idea that appears in several works [23], [24]. Lai and Fox [25] combine scans from real objects with models from Google 3D Warehouse to create an initial training set. In our approach, we solely base our classifier on synthetic models, and use those for getting object poses and to verify detection. Additionally, we show how our classification results can be combined from multiple scans to improve the results.

III. 3D Point Cloud Segmentation

Our classification of objects is based on the detection of the different parts that compose them. To determine these parts, we segment the 3D point clouds representing the objects and scenes. A segmentation defines a disjunct partition $\mathcal{P} = \{S_1, \ldots, S_M\}$ of the 3D point cloud. Our segmentation method follows a criterion based on a maximum angle difference between the surface normals. This condition is easily checked and can be applied to any type of surface. For each point, we calculate its normal by robustly identifying a tangent plane at the selected point and approximating the point's neighborhood (inside a radius of 3 cm) using a height function relative to this plane, in the form of a 2^{nd} order bi-variate polynomial defined in a local coordinate system [26]:

$$h_{(u,v)} = c_0 + c_1 u + c_2 v + c_3 uv + c_4 u^2 + c_5 v^2, \qquad (1)$$

where u and v are coordinates in the local coordinate system lying on the tangent plane. To obtain the unknown coefficients c_i, we perform a direct weighted least squares minimization and project the point onto the obtained surface. By choosing the query point to be at the origin of the local coordinate system ($\vec{U} \perp \vec{V} \perp \vec{N}$, with $\vec{U}$ and $\vec{V}$ in the plane, and $\vec{N}$ parallel to its normal), we can easily compute the normal $\vec{n}$ of the estimated surface by computing the two partial derivatives at $(0, 0)$ and the cross product $\vec{n} = (\vec{U}+c_1\vec{N})\times(\vec{V}+c_2\vec{N})$. The surface normals get more and more accurate as the order of the fitted polynomial increases, but in our experiments we found an order of 2 to give sufficiently good results for segmentation while keeping the computational costs low.

Using the obtained normals for each point in the cloud, we apply a region growing algorithm where we mark a point p as belonging to a part S if the distance between the point p and some point in S is closer than $\delta = 5$ cm, and if the angle formed by the normal of p and the seed normal of S is less than $\alpha = 40°$. Seed points are iteratively selected as points with the lowest curvature that do not belong to any part yet. This ensures that flat parts are identified first and makes the identification process more robust. The parts that have less

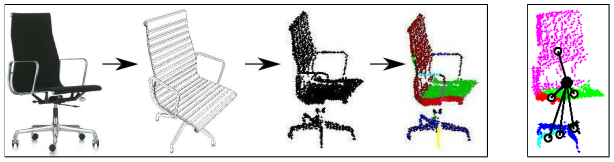


Fig. 2. Left: Example point cloud acquisition and segmentation of a chair. Right: Example shape model for a partial view. Please note that one of the legs and the extensible beam of the chair were occluded in the scan.

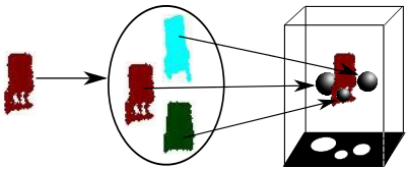


Fig. 3. Example word activation and corresponding 2D voting space.

than 10 points are considered to be too small, and are most probably produced in regions with high normal variations or by spurious points. Thus we perform a simple distance-based region growing to group them together and the resulting parts that are still too small are discarded. The parameters were selected because they provide good partitions in our setup (see Sect. VII). An example segmentation of an object is depicted in Fig. 2, and segmentations of point clouds in indoor environments are shown in Fig. 4.

Finally, our segmentation method produces parts with only slight curves. As explained in the introduction, the reason is that this kind of surface is very common in furniture objects in indoor environments. However, cylindrical objects would be broken up into parts covering at most 2α degrees, and the extracted features account for the curvature of the part.

IV. Training Objects from Web Catalogs

As explained in the introduction, the goal of this work is to develop a system that allows robots to query object databases in the Web to obtain information about typical objects found in indoor environments. In this work we use established Web databases of objects. In particular, we download CAD models from Google 3D Warehouse [27], Vitra's Furnish.net database for office equipment [28], and EasternGraphics' web catalogs [29].

To obtain realistic point cloud representations for these objects, we simulated our laser scanner's sweeping motion on the robot, intersected each beam with the CAD model of the object, and added realistic noise to the depth measurements. Each obtained scan was additionally segmented using the algorithm described in Sect. III. An example process for obtaining training data for a chair is shown in Fig. 2. The whole process takes, on average, 4.67s per view on a single core using a good graphics card.

V. Vocabulary of Parts

We build a common vocabulary of parts for all the classes in the training data, since most of the objects contain similar parts. The vocabulary is constructed by segmenting the training objects using the algorithm from Sect. III. Each part is then represented by a feature vector encoding its geometrical properties. Finally, the feature vectors are clustered.

A. Feature Vectors for Parts

For each part S obtained in the segmentation from Sect. III, we calculate the following set of geometrical features:

1) Proportion of boundary points in S computed as in [30].
2) Average curvature of S computed as the smallest eigenvalue's proportion to the sum of eigenvalues in the local neighborhoods of all points.
3) Volume occupied by the voxelized points in S.
4) We calculate three eigenvalues, e_1, e_2, e_3, of S and calculate six proportions: e_1/e_2, e_2/e_3, e_1/e_3, e_1/sum, e_2/sum, e_3/sum, where $sum = e_1 + e_2 + e_3$.
5) We obatin three eigenvectors, $\vec{e_1}$, $\vec{e_1}$, $\vec{e_1}$, of S, project the points onto each of them, and calculate three metric variances v_1, v_2, v_3 (which we used instead of e_1, e_2, e_3).
6) Orientation of the eigenvector corresponding to the smallest eigenvalue, indicating the orientation of S.
7) We project the points onto each eigenvector and get the distance to the farthest point from the medium in both directions $l^1_{\vec{e_1}}$, $l^2_{\vec{e_1}}$, $l^1_{\vec{e_2}}$, $l^2_{\vec{e_2}}$, $l^1_{\vec{e_3}}$, $l^2_{\vec{e_3}}$. We then calculate the following values: $(l^1_{\vec{e_1}} + l^2_{\vec{e_1}})$, $(l^1_{\vec{e_2}} + l^2_{\vec{e_2}})$, $(l^1_{\vec{e_3}} + l^2_{\vec{e_3}})$, $(l^1_{\vec{e_1}}/l^2_{\vec{e_1}})$, $(l^1_{\vec{e_2}}/l^2_{\vec{e_2}})$, $(l^1_{\vec{e_3}}/l^2_{\vec{e_3}})$, $(l^1_{\vec{e_1}} + l^2_{\vec{e_1}})/(l^1_{\vec{e_2}} + l^2_{\vec{e_2}})$.
8) Three proportions between features 5 and 7: $v_1/(l^1_{\vec{e_1}} + l^2_{\vec{e_1}})$, $v_2/(l^1_{\vec{e_2}} + l^2_{\vec{e_2}})$, $v_3/(l^1_{\vec{e_3}} + l^2_{\vec{e_3}})$.
9) Proportion between the occupied volume (feature 3) and the volume of the oriented bounding box of the part.

Each part S is finally represented by a vector containing 24 features normalized to the range [0,1].

B. Words and Shape Models

The resulting set of training feature vectors is clustered to obtain the words forming the vocabulary of parts. In our approach, we apply k-means, since it has given good results in previous works [31], [32]. After applying k-means to the training feature space, we obtained a clustering $\mathcal{C} = \{C_1, \ldots, C_V\}$, which represents our vocabulary of parts. Each cluster C_i is called a *word*. An example word is shown in the center of Fig. 3 representing the back of a chair from different views.

In addition, and following [16], we learn a shape model for each training object view. This model specifies the distance relations among the different parts that compose the object. We extract the center of mass $s \in \Re^3$ of each part S, and the center of mass $p \in \Re^3$ of the complete point cloud. We then calculate each relation as the 3D vector $\vec{d} = p - s$. An example is shown in the right image of Fig. 2.

VI. Object Recognition

In contrast to previous works [11], [12], we do not isolate possible objects in the scene before the classification. In our approach, we detect the objects by simultaneously collecting the information provided by all the parts in the scene.

Our object recognition process is composed of three main steps: i) a set of hypotheses is generated which indicate

possible locations for objects in the point cloud, ii) a selection of the best hypotheses is done following a set of criteria, iii) model fitting and verification is applied.

A. Hypotheses Generation

Given an initial partition $\mathcal{P} = \{S_1, \ldots, S_M\}$ of a 3D point cloud representing an indoor environment, we first obtain the corresponding feature vector f_i for each part S_i. Each feature vector is then matched to a subset of words $\mathcal{A} \subset \mathcal{C}$ from the learned vocabulary (activation), which constitute possible interpretations of the part S. Each element of an activated word casts a vote for a possible object location. This scheme is known as probabilistic Hough voting [16], and an example is shown in Fig. 3. In this case, a part in the scene activates a word representing backs of chairs. Each element in A casts a vote for the center of the chair inside the point cloud.

Formally, and following [16], given a feature vector f representing a part S located at position l in a 3D point cloud, the probability of finding an object of class o at position x is

$$p(o, x \mid f, l) = \sum_i p(o, x \mid A_i, f, l) p(A_i \mid f, l). \quad (2)$$

The term $p(o, x \mid A_i, f, l)$ represents the probability of finding object o at location x given that feature f at position l activated the word A_i. The term $p(A_i \mid f, l)$ indicates the probability that the feature vector f matches the word A_i.

Based on [16] we make two further assumptions. First, the probability of finding an object given an activated word A_i is independent of the feature vector f. Second, the activation of a word A_i by feature vector f is independent of the location of the corresponding part S. Thus,

$$p(o, x \mid f, l) = \sum_i p(o, x \mid A_i, l) p(A_i \mid f). \quad (3)$$

It remains to describe each term in (3). The term $p(A_i \mid f)$ represents the activation of A_i by f. Following [16] we do not activate just one vocabulary word, but several of them. Each activation is assigned a probability indicating how well the feature vector f matches the codebook entry A_i as

$$p(A_i \mid f) = \frac{\mathrm{w}(f, A_i)}{\sum_j \mathrm{w}(f, A_j)} \quad (4)$$

where $\mathrm{w}(f, A_i)$ is a distance function proportional to the inverse of the Euclidean distance between vector f and the mean vector of cluster A_i.

The term $p(o, x \mid A_i, l)$ represents the probability of finding object o at position x given the activated word A_i at location l. Once a word is activated, each of its elements a casts a vote to the position $x_v = l + \vec{d}$. Here l is the position of the part S found in the point cloud, and $\vec{d}$ is the relation vector of the element's shape model (see Sect. V-B). Finally the probability $p(o, x \mid A_i, l)$ is given by

$$p(o, x \mid A_i, l) = \begin{cases} \frac{1}{\#(A_i, o)} & \text{if } x = x_v \text{ ;} \\ 0.0 & \text{otherwise,} \end{cases} \quad (5)$$

where $\#(A_i, o)$ indicates the number of elements a in word A_i that pertain to object o.

B. Hypotheses Selection

According to [16], the final score $V(o, x)$ of a hypothesis representing an object and its position can be obtained by marginalizing over all the parts found in the scene,

$$V(o, x) = \sum_k p(o, x \mid f_k, l_k) p(f_k \mid l_k) \; . \quad (6)$$

The first term in this expression is calculated as in (3), and the second term $p(f_k \mid l_k)$ is assumed to be a uniform distribution.

Following the approach in [16], the different object hypotheses are found as local maxima in the voting space using a search window whose size corresponds to the width of the particular object class we are looking for. We project the votes onto the (x,y)-plane in order to simplify this search (see Fig. 3). After obtaining the local maxima, we apply a threshold to select the best hypotheses. Moreover, hypotheses are checked for inconsistencies in the space. In our case, two hypotheses for the same type of object at two different locations x_i and x_j are inconsistent if the 2D-convex hulls of the objects centered at x_i and x_j overlap. This condition assumes that all objects are lying on the floor.

C. Model Fitting and Verification

Having detected positions where a certain object type is likely to be found, we verify the detection and select the best model for the object, along with its pose. To be able to do this, we first need to retrieve the most important parts that voted for the location. Some of these parts are incorrect but their weights are low.

Our task is now to select the model and its pose from a 3D model database that explains most of the detected object parts. We employ a more detailed model database for this, in order to be able to account for the variations in shapes and sizes of the different objects. To fit these models to our 3D scans robustly and efficiently, we transform them to noiseless point clouds by filling the outer triangle faces with points, and use a RANSAC-based algorithm to select the best pose and the best model for each detected object. Since we take the complete model into account at once instead of each view separately, we can reduce the number of trials considerably and we do not require the point cloud to come from a single scan. This can become important later, as detailed in Sect. VII-C. The search time is further reduced by assuming that objects are upright, thus only the location in 2D and the rotation around the up axis is required to be found.

In a regular RANSAC algorithm, one can estimate the number of required iterations T to find the best model with probability $p_{success}$ as

$$1 - p_{success} = (1 - p_{good})^T \Rightarrow T = \frac{log(1 - p_{success})}{log(1 - p_{good})}. \quad (7)$$

The value of $p_{good} = w^n$ is the probability of randomly selecting the best model, and can be estimated after each fit (that is better than the best one found so far) as the probability w of selecting a good sample to the power of the number of samples n. As the algorithm finds models that are increasingly

better, the value of w can be updated as $\#inliers/\#points$, resulting in a decrease in the number of iterations needed. Thus the number of iterations adapts to the estimated number of matches, starting at infinity when no inliers are known and decreasing to a number that ensures that the chance of finding a better model than the current one drops below $1 - p_{success}$.

As the runtime depends on the number of samples, it is advisable to keep it as low as possible. If we were to pick both the model and the scan points at random, we would have had to use $p_{good} = (\#correct\_matches/\#possible\_matches)^2$ a very small number that is also hard to estimate. However, if we assume that by selecting a random point from the scan, we can find the corresponding model point. This simplifies the equation to $p_{good} = \#covered\_scan\_points/\#all\_scan\_points$. The number of covered scan points can be found by nearest neighbor searches with a maximum distance in a search tree.

We found that selecting the corresponding model point to a scan point can be done by iterating over the possible matches (points at similar height), and selecting the one that would result in a transformation that covers most scan points. Significant speedups can be achieved by selecting only a subset of the possible correspondences and scan points. Our experiments provided good results with around 50% of the points checked. Additionally, by keeping track of the best score found, subsequent searches can be stopped early if it becomes clear that they can not produce better scores. The same principle can be applied over multiple models as well.

Using these techniques, we were able to identify the best pose of the good model in around 15s on a standard dual-core laptop, without parallelization or other optimizations. Our models contain between 5000-15000 points, while the identified object parts around 2500. For non-optimally matching models, the verification takes around 60s, while non-matching models are rejected in under a second. After each object model from the identified category is fitted to the object parts, we select the one that explains the most model points.

Before this model and its pose can be used for various applications, it first needs to be checked in order to filter out false positives. We first check if the model covers less than 50% of the points of the parts, and reject such detections (e.g. the chair in column two of Fig. 8). The threshold of 50% is tied to how well the CAD models and the real objects match. The real chair and the best fitting model are not identical; however, all CAD models we used cover much more than 50% of the points if a good match is found.

We then check if the identified pose contradicts the measurements, i.e. if it has faces that should have been visible while scanning, but the laser returned points behind them. This can be done efficiently by building one occupancy grid per scan, in which each voxel is labeled as free, occupied or unknown [33]. We reject models that have large parts in free space, e.g. the sideboard in the first column of Fig. 8. Such a grid is needed for the operation of the robot for tasks like collision avoidance and path planning. Thus it is natural to use it for the verification of model fittings attempts, as we did in our experiments involving smaller objects as well [20].

Finally, when the scan point has a normal vector which is close to being parallel to the upright axis, the rotation of the model can not be accurately estimated. In these cases, we use a random rotation and since we check a high percentage of points, the models are correctly identified. Tables are especially affected by this, and we see larger fitting errors in finding them but the best models get identified correctly and a subsequent fine registration using ICP [34] or one of its many variants can be used to correct the pose.

VII. Experiments

The goal of the experiments is to demonstrate that a robot can learn models for 3D office objects found on the Web, and using these models, is able to locate and categorize pieces of furniture in 3D point clouds representing indoor environments.

To carry out the experiments, we first downloaded the CAD models of a chair, a table and a sideboard (colored as blue, red and yellow in Fig. 1). These objects were obtained from different Web catalogs. Following the procedure of Sect. IV, we placed the objects in front of the simulated laser and rotated them around the up axis, obtaining 16 scans of each object which were segmented using our algorithm. A final common vocabulary of parts was created with $k = 40$. These values provide a good balance between performance and computational costs (see Fig. 6).

To obtain point clouds from real scenes, we used a Hokuyo laser mounted on a pan tilt on a robot approximately 1.5m above the floor (see Fig. 1). Service robots working in human environments typically have their 3D sensor at that height such that its perspective is similar to that of a person. Both of the robots we used had such a setup, and since they both ran ROS (www.ros.org), changing the robot was transparent to our algorithm. However, different robots or sensors at different heights can be used if the scanning of the training data is adapted accordingly. Finally, the point clouds were segmented using the same algorithm as for the training data. Each obtained part activated the best two vocabulary entries.

A. Recognition of Objects in Real Environments

The first experiment was carried out in an office of our floor at TUM. The corresponding segmented 3D point cloud is shown in Fig. 4. This office contained two tables, different armchairs and a sideboard. We would like to point out that the objects located in the office were similar to but not the same as the 3D models we downloaded from the Web. In particular, the table in the middle of the chairs was round and we did not have any round table in our training set, and the chairs around the table were different from the training models. This setting aimed to demonstrate that our approach is well-suited for detecting categories of objects instead of concrete instances only.

We set the thresholds for the final hypotheses to 1.0 for tables, 0.6 for chairs, and 1.4 for sideboards. These values were selected as desired working points in classification performance plots (Fig. 5). The result of applying our detector is shown in Fig 4. The two tables were correctly detected and the

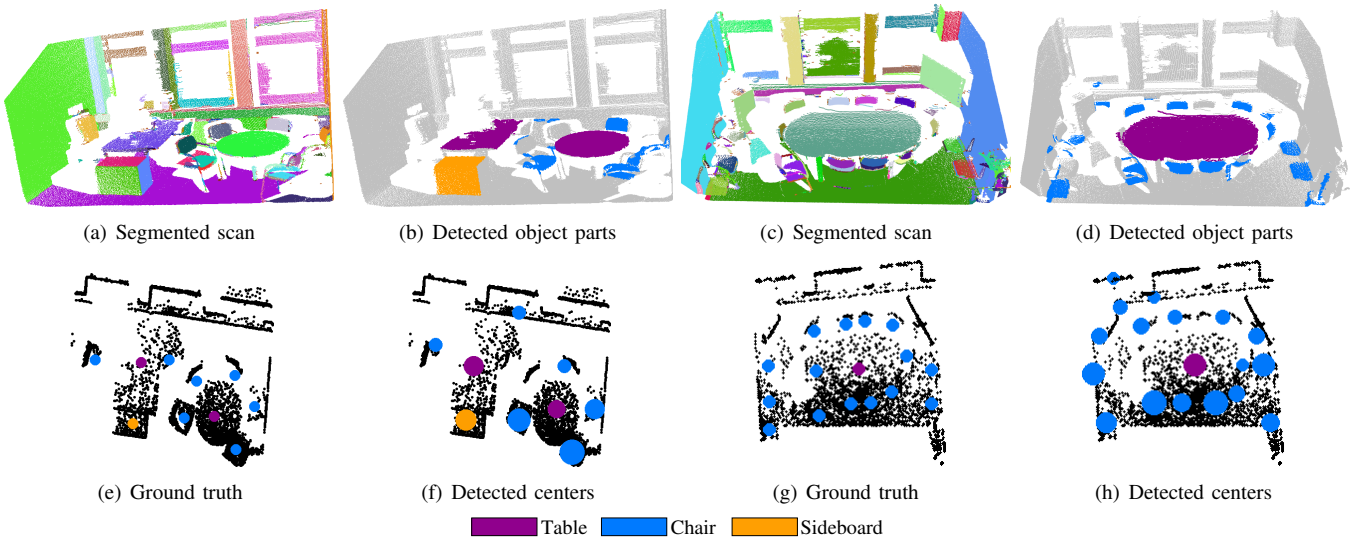
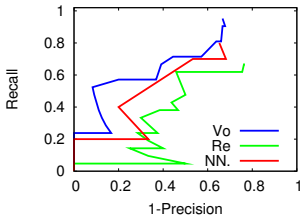


Fig. 4. The segmented scans of an office and a seminar room are depicted in (a) and (c) respectively. The detections of the objects' parts for the office are shown in (b), and for the seminar in (d). In (e) and (g) we can see the ground truth of the positions for the objects' centers in the office and seminar room (bird's eye view). The detected centers obtained by our method are shown in (f) and (h), where bigger circles indicate a bigger weight. *Best viewed in color.*



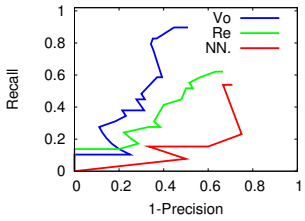
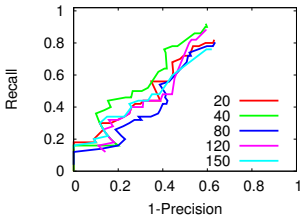


Fig. 5. Classification results for the office (left) and seminar (right) using our approach (Vo), nearest neighbor (NN), and a reduced set of features (Re). More details are given in the main text. *Best viewed in color.*



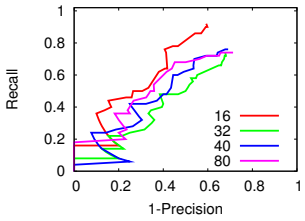


Fig. 6. Left: Classification performance using different number clusters k and a fixed number of 16 views. Right: Classification performance using different number of views and fixed number of clusters k=40. *Best viewed in color.*

system was able to detect 5 chairs out of 7, although including one false positive. An additional experiment was carried out in a seminar room containing a big table and 15 chairs around it. The resulting classifications are also shown in Fig. 4.

Data acquisition using our laser takes 10 s. Smoothing, segmenting and extracting the features takes another 7.12 s on average using C++ code on a single core. The time needed to select the final hypotheses for an object center mainly depends upon the total number of parts in the scene and the number of clusters in the vocabulary (since each of the parts should activate the closest clusters). It also depends on the number of final selected hypotheses because they are checked for inconsistencies (see Sect. VI-B). Using our current configuration, and without any optimization, our approach took 0.20 s for the office, and 0.24 s for the seminar using Matlab on an Intel Core i5.

To quantitatively evaluate the performance of our approach, we generated recall vs (1-precision) plots for the two previous scenes. Different false positive values were obtained by increasing the threshold for the final hypotheses selection. A true positive is obtained when the distance between the detected center and the original center is less than half of the width of the object. The test set included two scans of the office and two scans of the seminar obtained from different viewpoints. The resulting plots are shown in Fig. 5. In each plot we additionally compare our approach based on a vocabulary of parts (Vo) with two simplified versions: a version in which only the nearest neighboring part in the training data is used for voting (NN); and a version with a reduced number of features (ignoring features 4, 7 and 8) (Re). This last version aims to show the importance of all the features in the final classification. In both scenes our approach outperforms the simplified ones.

We also analyzed the influence of the vocabulary size in the classification performance. Results are shown in the left image of Fig. 6. Finally, in the right image of Fig. 6 we analyzed the performance of the classifier when increasing the number of training views. Interestingly, using 16 views for training gives slightly better results than using more. This could be because the segmentation and features produce similar patches. Improving the feature extraction is on our agenda, thus exploiting our classification to the fullest. However, the results are quite similar for the different parameters.

B. Model Fitting and Verification

In this section we present results on applying model fitting for the final verification of the detected objects. In the first experiment we used the classification results obtained in the previous section for the office and seminar environments. The classification outputs consist of the center of the detected objects together with the parts that contributed to its detection (see Fig. 4). Using these outputs we fitted the best original CAD model for each final hypothesis using the corresponding retrieved parts and verifying the fitting as described in Sect. VI-C. The results are shown in Fig. 7.

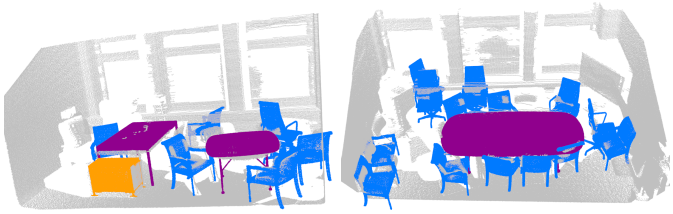


Fig. 7. Model fitting in the scenes from Fig. 4.

The poses are good for most of the objects and a subsequent ICP step would improve them further. However, here we focus on the possibility of using it to reject false positives. The two false positives in the right side of the seminar room were indeed removed, but the four chairs in the walls of the office and seminar room could not be filtered, as they covered the parts (which were quite small) well and were in occluded space. These could be filtered for example by having a 2D floor map available, as they are outside the boundaries of the rooms. False positive rejection is also shown in Fig. 8.

There were four occasions where the fitted models were incorrect. One chair in the office and one in the seminar room are oriented backwards because their seats were occluded and the best matching CAD model that was available fits the data best when it is rotated in the wrong direction. Because of the large occlusions, these orientations could not be filtered. Two other chairs in the left part of the scans had very small parts (31 and 32 points), rendering correct matching impossible. In the office the bad orientation was preserved due to the high occlusion, but in the seminar room it was rejected.

C. Recognition Using Multiple Views

In the following experiment we showed how to take advantage of multiple views of the same scene to alternatively improve the final detection of objects. We used the two scenes of Fig. 8 and applied the voting approach on each of the scenes as in the previous experiment. We then translated the resulting voting spaces into a common coordinate system. This process is equivalent to the alignment of the two scenes and the simultaneous voting in both of them. As shown in Fig. 9, after merging both voting spaces the system is able to reject both false positives detections: the sideboard sharing place with the chair in the first scene; and the shadow of the man in the second scene (see previous experiment). The improvement in the recognition is a consequence of the simultaneous accumulation of evidence from both views. Correct detected objects received a much higher weight since they are detected twice, whereas false positives objects get evidence from only one view.

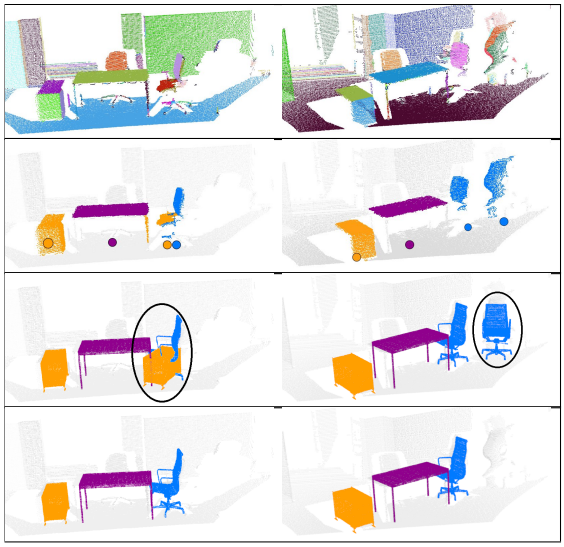


Fig. 8. Two examples of the whole process (see legend in Fig. 4).

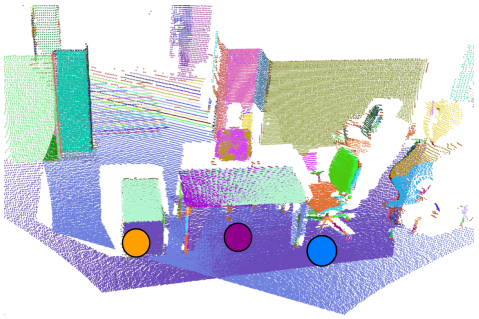


Fig. 9. The two scenes from Fig. 8 are shown in a common coordinate system, along with the detected object centers after merging the corresponding voting spaces (see legend in Fig. 4 for center colors).

VIII. Conclusions and Future Work

In this paper, we have presented an approach that allows a robot to learn models of office objects downloaded from the Web and to use them to detect and localize instances of these types of objects in new scenes represented by 3D point clouds.

We believe that our method is a step in the right direction – a step towards using a WWW for robots in a scalable way. In addition, the set of possible models can be limited in multiple ways, even obtaining a list of furniture pieces for a given environment is conceivable. Our next steps will be to extend the method for more object types, and to take advantage of the possibility of multiple segmentations per scan to increase the accuracy. Applying this approach to smaller objects with six degrees of freedom is possible, as the parts can be learned in a rotationally invariant manner. However, the effectiveness of the probabilistic voting in 3D and the computational complexity of precise model fitting remains to be tested.

Nowadays, typical indoor objects like chairs, tables or sideboards are found all over the world and globalization makes it likely to find the same objects in different places. The ultimate goal would be to enable a robot to download 3D models of objects from the WWW and to learn general representations of them in order to be able to recognize similar objects in a new environment somewhere else in the world. Taking the idea one step further, robots could also map their environments, and the point clouds corresponding to high-scoring detections could be used for training and verification or could even be shared with other robots, thus enriching existing databases by distributed world modeling.

What is needed, however, is the initial set of training examples, in an easily accessible way. As previously pointed out by other authors [25], one of the main problems when working with 3D point clouds is the availability of labeled training data. Model representation needs to be enriched as well, to account for different measurement units and orientations. Not to mention the problems arising from different data formats. Fortunately, there are efforts to solve these issues, and additional data can be obtained by shape morphing – e.g. using techniques such as the one from Hillenbrand [35].

In conclusion, by taking advantage of the explosion in the number of WWW resources that is driven by industry, we can avoid instrumentation to a large degree and allow robots to act successfully in unknown environments.

ACKNOWLEDGMENTS

This work was supported by the DFG excellence initiative research cluster CoTeSys, by Grants-in-Aid for JSPS Fellows number 22-00362, and by the PR2 Beta Program of Willow Garage, Menlo Park, CA. We would also like to thank our colleague Dejan Pangercic for his valuable support, and Vitra Furnishnet for granting us access to their model database.

REFERENCES

[1] R. B. Rusu, Z. C. Marton, N. Blodow, A. Holzbach, and M. Beetz, "Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, USA, 2009.

[2] D. Kragic and M. Vincze, "Vision for robotics," *Found. Trends Robot*, vol. 1, no. 1, pp. 1–78, 2009.

[3] A. Johnson, "Spin-images: A representation for 3-D surface matching," Ph.D. dissertation, Carnegie Mellon University, USA, 1997.

[4] R. Triebel, R. Schmidt, O. M. Mozos, and W. Burgard, "Instace-based AMN classification for improved object recognition in 2D and 3D laser range data," in *Int. Joint Conference on Artificial Intelligence*, India, 2007, pp. 2225–2230.

[5] P. A. de Alarcon, A. D. Pascual-Montano, and J. M. Carazo, "Spin images and neural networks for efficient content-based retrieval in 3D object databases," in *Int. Conf. on Image and Video Retrieval*, 2002.

[6] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "NARF: 3D range image features for object recognition," in *IROS Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics*, 2010.

[7] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of markov random elds for segmentation of 3D scan data," in *Conf. on Computer Vision and Pattern Recognition*, 2005.

[8] R. Triebel, K. Kersting, and W. Burgard, "Robust 3D scan point classification using associative markov networks," in *Int. Conf. on Robotics and Automation*, 2006.

[9] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Europ. Conf. on Computer Vision*, 2004.

[10] Z. Wu, Y. Wang, and G. Pan, "3D face recognition using local shape map," in *IEEE Int. Conf. on Image Processing*, 2004.

[11] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "Accurate object localization in 3D laser range scans," in *Int. Conf. on Advanced Robotics*, USA, 2005, pp. 665–672.

[12] K. Klasing, "Aspects of 3D perception, abstraction, and interpretation in autonomous mobile robotics," Ph.D. dissertation, Technische Universität München, 2010.

[13] D. Huber, A. Kapuria, R. R. Donamukkala, and M. Hebert, "Parts-based 3D object classication," in *Conf. on Computer Vision and Pattern Recognition*, 2004.

[14] S. Ruiz-Correa, L. G. Shapiro, and M. Meila, "A new paradigm for recognizing 3-D object shapes from range data," in *Int. Conf. on Computer Vision*, 2003.

[15] S. Agarwal and D. Roth, "Learning a sparse representation for object detection," in *Europ. Conference on Computer Vision*, 2002.

[16] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 259–289, 2008.

[17] M. Sun, B. Xu, G. Bradski, and S. Savarese, "Depth-encoded hough voting for joint object detection and shape recovery," in *Europ. Conference on Computer Vision*, 2010.

[18] Z. C. Marton, R. B. Rusu, D. Jain, U. Klank, and M. Beetz, "Probabilistic Categorization of Kitchen Objects in Table Settings with a Composite Sensor," in *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, USA, October 11-15 2009.

[19] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, June 2007.

[20] Z.-C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, "General 3D Modelling of Novel Objects from a Single View," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taiwan, 2010.

[21] A. S. Mian, M. Bennamoun, and R. A. Owens, "Matching tensors for automatic correspondence and registration," in *Europ. Conf. on Computer Vision*, 2004.

[22] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *IEEE Int. Conf. on Robotics and Automation*, Japan, 2009.

[23] H.-P. Chiu, L. Kaelbling, and T. Lozano-Perez, "Virtual training for multi-view object class regnition," in *IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, 2007.

[24] A. Toshev, A. Makadia, and K. Daniilidis, "Shape-based object recognition in videos using 3d synthetic object models," in *IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, 2009.

[25] K. Lai and D. Fox, "3D laser scan classification using web data and domain adaptation," in *Robotics: Science and Systems*, USA, 2009.

[26] Z. C. Marton, R. B. Rusu, and M. Beetz, "On Fast Surface Reconstruction Methods for Large and Noisy Datasets," in *IEEE Int. Conf. on Robotics and Automation*, 2009.

[27] *Google 3D Warehouse*. [Online]. Available: http://sketchup.google.com/3dwarehouse/

[28] *Vitra Furnishnet*. [Online]. Available: http://www.vitra.com

[29] *EasternGraphics*. [Online]. Available: http://portal.pcon-catalog.com

[30] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D Point Cloud Based Object Maps for Household Environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[31] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[32] A. Coates, H. Lee, and A. Y. Ng., "An analysis of single-layer networks in unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[33] N. Blodow, R. B. Rusu, Z. C. Marton, and M. Beetz, "Partial View Modeling and Validation in 3D Laser Scans for Grasping," in *IEEE-RAS Int. Conf. on Humanoid Robots*, France, 2009.

[34] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, February 1992.

[35] U. Hillenbrand, "Non-parametric 3D shape warping," in *Proc. Int. Conf. on Pattern Recognition*, 2010.